Unidade VI: Ordenação Interna – Comparação dos métodos

PUC Minas

Número de Comparações

Método	Complexidade
Inserção	$O(n^2)$
Seleção	$O(n^2)$
Bolha	$O(n^2)$
Quicksort	$O(n \log(n))$*
Mergesort	$O(n \log(n))$
Heapsort	$O(n \log(n))$

Tempo de Execução

- O método que levou menos tempo real para executar recebeu o valor 1 e os outros receberam valores relativos
- Elementos em ordem aleatória

	5.00	5.000	10.000	30.000
Inserção	11,3	87	161	–
Seleção	16,2	124	228	–
Quicksort	1	1	1	1
Heapsort	1,5	1,6	1,6	1,6

Tempo de Execução

- O método que levou menos tempo real para executar recebeu o valor 1 e os outros receberam valores relativos
- Elementos em ordem crescente

	500	5.000	10.000	30.000
Inserção	1	1	1	1
Seleção	128	1.524	3.066	–
Quicksort	4,1	6,3	6,8	7,1
Heapsort	12,2	20,8	22,4	24,6

Tempo de Execução

- O método que levou menos tempo real para executar recebeu o valor 1 e os outros receberam valores relativos
- Elementos em ordem decrescente

	500	5.000	10.000	30.000
Inserção	40,3	305	575	–
Seleção	29,3	221	417	–
Quicksort	1	1	1	1
Heapsort	2,5	2,7	2,7	2,9

Inserção

- É o mais interessante para arquivos pequenos (com menos de 20 elementos), podendo ser mais eficiente do que algoritmos que tenham comportamento assintótico mais eficiente.
- O método é estável.
- Possui comportamento melhor do que o método da bolha que também é estável.
- Sua implementação é tão simples quanto as implementações do Bolha e Seleção.
- Para arquivos já ordenados, o método é O(n)
- O custo é linear para adicionar alguns elementos a um arquivo já ordenado.

Seleção

- É vantajoso quanto ao número de movimentos de registros, que é O(n)
- Deve ser usado para arquivos com elementos muito grandes, desde que o número de elementos a ordenar seja pequeno

Quicksort

- É o algoritmo mais eficiente que existe para uma grande variedade de situações
- Pior caso realiza $O(n^2)$ operações
- O principal cuidado a ser tomado é com relação à escolha do pivô
  - A escolha do elemento do meio do arranjo melhora o desempenho quando o arquivo está total ou parcialmente ordenado
  - O pior caso tem uma probabilidade muito pequena de ocorrer quando os elementos forem aleatórios
  - Geralmente se usa a mediana de uma amostra de três elementos para evitar o pior caso
- Usar inserção em partições pequenas melhora desempenho significativamente

Mergesort

- Seu custo é sempre de O(n log(n)) operações
  - Deve ser considerado quando alto custo de pior caso não pode ser tolerável.
- Não varia com a entrada (não é adaptável)
- É estável
- Requer espaço extra de memória proporcional a n.

Heapsort

- É um método de ordenação elegante e eficiente
- Não necessita de nenhuma memória adicional
- Executa sempre em tempo proporcional a $O(n \log(n))$
- Aplicações que não podem tolerar eventuais variações no tempo esperado de execução devem usar o heapsort

Referência

- ZIVIANI, Nivio. Projeto de algoritmos: com implementações em Java e C++. São Paulo: Cengage Learning, c2007.